Implementação de Hierarquia de Memória Cache em VHDL

Andrei Massaini

1 Implementações e Estrutura

1.1 Cache com Mapeamento Direto

A implementação da cache com mapeamento direto foi realizada com os seguintes parâmetros:

- **Configuração**: 32 bits de endereço, 256 linhas, 32 bits por word.
- **Divisão do endereço**: Tag (22 bits), Índice (8 bits), Offset (2 bits)
- **Estrutura**: Arrays para dados, tags e bits de validade

Para verificar se houve hit, a tag é comparada com a tag que está armazenada no índice a ser comparado da cache:

Listing 1: Verificação de hit na Cache com Mapeamento Direto

```
-- Verificando se houve hit:
hit_v := '0';
if (valid_bits(index_v) = '1') and (cache_tags(index_v) = tag_v) then
    hit_v := '1';
end if;
```

A implementação escolhida para manter a coêrencia dos valores foi a de write-troguth, ou seja a MP era atualizada no mesmo instante que a cache.

1.2 Cache com Mapeamento Associativo de 4 Vias

A implementação da cache associativa apresenta os seguintes parâmetros:

- **Configuração**: 32 bits de endereço, 64 conjuntos × 4 vias, 32 bits por palavra
- **Divisão do endereço**: Tag (24 bits), Índice (6 bits), Offset (2 bits)
- **Estrutura**: Arrays bidimensionais para dados, tags, bits de validade e contadores LRU

A verificação de hit requer a comparação da tag com todas as vias do set indexado:

Listing 2: Verificação de hit na Cache Associativa

```
-- Verificar hit na cache
hit_found := false;
hit_way := 0;
for i in 0 to WAYS-1 loop
    if valid_bits(index_v, i) = '1' and cache_tags(index_v, i) = tag_v then
        hit_found := true;
        hit_way := i;
        exit;
    end if;
end loop;
```

Esta implementação permite duas políticas distintas de substituição, selecionáveis durante a execução.

2 Políticas de Substituição

2.1 LRU (Least Recently Used)

A política LRU mantém um histórico de uso para identificar a via menos recentemente acessada:

- Utiliza contadores de 2 bits para cada via em cada conjunto
- Via com valor "00": mais recentemente usada (MRU)
- Via com valor "11": menos recentemente usada (LRU)

Listing 3: Implementação da Política LRU

```
-- Atualizar contadores LRU
for i in 0 to WAYS-1 loop
    if i = hit_way then
        lru_counters(index_v, i) <= "00";  -- MRU
    elsif lru_counters(index_v, i) < "11" then
        lru_counters(index_v, i) <= lru_counters(index_v, i) + 1;
    end if;
end loop;

-- Selecionar via para substitui  o
lru_way := 0;
for i in 1 to WAYS-1 loop
    if lru_counters(index_v, i) > lru_counters(index_v, lru_way) then
        lru_way := i;
    end if;
end loop;
```

2.2 Random

A política Random oferece uma alternativa mais simples:

- Utiliza contador circular de 2 bits incrementado a cada ciclo
- Via selecionada = valor atual do contador (0-3)
- Implementação mais simples, requer menos hardware

Listing 4: Implementação da Política Random

```
-- Incrementar contador pseudo-aleat rio
random_count <= random_count + 1;

-- Selecionar via para substitui  o
empty_way := to_integer(random_count);
```

3 Comparação entre Implementações

Característica	Mapeamento Direto	Mapeamento Associativo
Complexidade	Simples, menos hardware	Complexa, mais hardware
Utilização	Menor eficiência, mais conflitos	Maior eficiência, menos conflitos
Latência	Menor (verificação única)	Maior (verificações paralelas)
Taxa de acertos	Inferior	Superior
Escalabilidade	Limitada por conflitos	Melhor para caches maiores

Tabela 1: Comparação entre Mapeamento Direto e Associativo

4 Resultados da Simulação

4.1 Cache com Mapeamento Direto

Os testes da cache com mapeamento direto revelaram:

- **Escritas iniciais**: 10 endereços distintos, todos com miss
- **Leituras subsequentes**: Hits nos mesmos endereços
- **Teste de conflito**: Escrita em endereço 1024 substituiu dados do endereço 0, demonstrando a limitação fundamental do mapeamento direto

4.2 Cache Associativa

Os testes da cache associativa demonstraram:

4.2.1 Política LRU

- **Teste de sobrecarga**: Escrita em 5 endereços (0, 256, 512, 768, 1024) mapeando para o mesmo conjunto causou substituição do endereço menos recentemente usado
- **Teste de localidade temporal**: Acesso repetido ao endereço 0 o protegeu de substituição quando novos endereços foram introduzidos

4.2.2 Política Random

- **Teste de sobrecarga**: Substituição ocorreu em vias imprevisíveis
- **Leituras subsequentes**: Padrão misto de hits e misses conforme esperado para seleção pseudo-aleatória

4.3 Análise Comparativa de Desempenho

Implementação	Acessos	Taxa de Acertos
Mapeamento Direto	25	40%
Associativo (Random)	25	60%
Associativo (LRU)	25	72%

Tabela 2: Comparação de desempenho entre implementações

5 Conclusão

A implementação e análise das diferentes arquiteturas de cache demonstram claramente o trade-off entre complexidade de hardware e desempenho. A cache associativa com LRU apresentou a melhor taxa de acertos (72%), seguida pela associativa com Random (60%) e pela cache de mapeamento direto (40%).

Em sistemas com restrições de área e energia, o mapeamento direto pode ser adequado para padrões de acesso previsíveis. Para aplicações com padrões complexos, o custo adicional da implementação associativa com LRU se justifica pelo aumento significativo na taxa de acertos, reduzindo o tempo médio de acesso à memória.

Possíveis otimizações futuras incluem a implementação de políticas write-back e a exploração de caches multinível para balancear melhor os trade-offs entre latência e taxa de acertos.